Growing and Expanding Transformational Leaders Team with Experiments

Hiroyuki Ito, SET taskforce, LINE, Japan

Since I joined LINE Corporation as the first member of “SET” (Software Engineer in Test) in 2017, we have been solving a variety of software and organizational problems.

Through these achievements, we have been adjusting our responsibilities from software quality to software delivery, profitability, and organizational processes.

This report is about why and how we are becoming a team of “Transformational Leaders” [1] that is responsible for software delivery performance and organizational culture [2] based on experiments.

# Introduction

"LINE" is a free chatting and telecommunication service for smartphones that has released since 2011. Our company name is derived from this service.

After the first release, LINE Corporation has been increasing users and messages transferred rapidly and globally. Especially, high sound quality with free, and the "sticker" feature that we can send a variety of rich emoticons as a message attracted a lot of users.

For adapting to the rapid growth of LINE, we have been improving LINE's architectures and code base iteratively. We chose “Microservice Architecture” [3] to earn scaling out, independent development, and fast delivery capabilities.

However, outages of LINE have also been increasing. Features, especially fintech ones like payments and banking, have been increasing dramatically. Troubles at "Integration Points" [4] among each Microservices have also been increasing. They mean increases of negative monetary impacts to LINE users.

For reducing outages and improving product development processes, LINE Corporation decided to open new positions for a Scrum Master, a DevOps engineer, and an SET. I joined LINE Corporation as the first member of SET in 2017.

# CLARIFY DEMANDS AND RESPONSIBILITIES TO START

## CHALLENGES

When I joined LINE Corporation, there were lots of problems, which relate to not only software quality but a variety of software and organizational problems.

The biggest challenge was the confusions and disagreements about SET among stakeholders. There were no clear objective, missions, and responsibilities of SET. Additionally, there were no shared understandings about SET. Therefore, I needed to clarify them at first.

Other big challenge was that I was a newbie of LINE Corporation and I didn't have enough knowledge of our services, architectures, technologies used, and so on.

Moreover, there were few leaders to solve problems, which affect more than one team and/or service. In other words, few leaders could act beyond silos. We have been widely adopting to Microservice Architecture. It was critical to overcome this problem for solving outages quickly and properly with fewer impacts to users.

## ACTIONS

For the smooth start of SET activities, I did the following actions.

### GATHER INFORMATION WITH PRODUCT DISCOVERY

To clarify objective, missions, and responsibilities of SET, I utilized the idea of "Product Discovery" [5] taught by David Hussman for gathering necessary information.

At first, I analyzed our services and products. I utilized static code analysis tool named "SonarQube" [6] to know the code coverage and technical debts for each service. I also added simple unit and integration test scripts to know behavior of the products. Test scripts are good for understanding software under test [7].

Next, I focused on analyzing "outage reports". "Outage reports" mean both postmortem meetings and published reports. They are a treasure-trove of information we need to solve. I was able to know causes of outages, impact on sales and profits, and problematic products through these reports. I understood that public APIs provided for external users, and **Sticker Shop**, were the most problematic products. Additionally, I found that reducing "MTTR" (Mean Time to Repair) would be an impactful solution as the first step.

Moreover, I talked stakeholders like developers, QA persons, Product Managers, senior managers, and executives to hear their concerns and troubles directly and beyond silos. Stakeholders' worries are also a treasure-trove of information to improve. Through these conversations, I understood that they had lots of non-verbalized problems. I also learned that verbalizing problems through direct and honest conversations is critical for discovering real needs, shared understanding, and collaborations beyond silos.

### ESTABLISH SOLUTIONS WITH ITERATIVE AND INCREMENTAL CONSENSUS

At the end of the first week I joined LINE Corporation, I built the first rough ideas of SET including objective, missions, responsibilities, solutions, and milestones based on gathered information I mentioned above. Additionally, I proposed them to "decision makers" like senior managers and executives.

I don't think that I can build the perfect solutions and agree on them with decision makers at once. I supposed that it would be preferable not only me but decision makers to continue proposing ideas, getting feedbacks, and improving proposal. Additionally, there were few persons who could lead "strategy formulation" in LINE Corporation at that time. Leading decision-making gave impacts to decision makers and it was good for attracting their interests in SET. Therefore, I chose to iterate build-propose-learn cycle weekly as "Iterative and Incremental Consensus" approach.

My first proposal was focusing on improving **Sticker Shop** due to frequency of outages, however I didn't define milestones. Through this approach, rough milestones were enough useful for decision makers to understand tasks, plan, and due date easily and quickly. Additionally, they also said that it was OK to update milestones if we knew additional information. Moreover, they taught me that **public APIs** were more important than **Sticker Shop** at that time from business perspective. On the other hand, they agreed on my idea that utilizing Test Automation for making failure detection faster and reducing MTTR was valuable as SET's responsibility. Through this approach, I could improve my proposal step by step. Finally, we agreed on the first solution and milestone within 45 days since I joined LINE Corporation.

### MANAGING IMPACTS BY PROVIDING RESULTS EVERY WEEK

In parallel with "Iterative and Incremental Consensus", I tried to "manage impacts" [8] constantly to coworkers and decision makers for attracting their interests in SET.

From the first week I joined LINE Corporation, I achieved something every week and shared them with coworkers and decision makers. Especially, I shared working software or executable one. Additionally, I showed results quantitatively beyond silos.

Here is a list of achievements for my first 10 weeks.

|  |  |
| --- | --- |
| Week | Achievements |
| Week 1 | Started writing test scripts to understand services/products |
| Week 2 | Proposed first idea of SET activities to decision makers |
| Week 3 | Built mechanism to run and report static code analysis regularly |
| Week 4 | Shared to developers how to build static code analysis |
| Week 5 | Proposed milestones of activities to decision makers |
| Week 6 | Agreed with proposals/milestones with decision makers |
| Week 7 | Collected information and tools of QA/Tests in one place |
| Week 8 | Implemented failure detection for public APIs |
| Week 9 | Guided to start regular meetings with developers and QAs |
| Week 10 | Started solving problems by developers step by step |

Figure 1. The list of achievements for my first 10 weeks as SET.

When I had been managing impacts, I utilized "3 KPIs"; Sales, Profit, and Employee Satisfaction. When I had worked at Rakuten, one of senior executives and my supervisor had taught me that every business can measure with these 3 KPIs. After that, I have been utilizing it for all activities.

Here are examples. I chose all activities as SET for improving Sales and Profit. I implemented test scripts for reducing MTTR, not only for expanding Test Automation. Additionally, I picked up actions that could affect Employee Satisfaction. For decision makers, I focused on discovering and verbalizing their anxieties, and providing quantitative information. For Developers and QAs, I tried to stimulate appetites for learning.

As a result, many colleagues started talking about SET. Their interests in SET led collaboration with product development teams, QAs/Test Automators, and Product Managers. It meant I could lead problem-solving beyond silos. Additionally, decision makers started supporting SET activities positively. Quick agreement on the first solution and milestones of SET was a good sign.

## RETROSPECTIVE

The idea of "Product Discovery" worked for clarifying responsibilities and activities of SET. Additionally, "Iterative and Incremental Consensus" was useful for collaborating with decision makers and agreeing with them quickly. Moreover, "managing impacts with 3 KPIs" attracted lots of colleagues from business perspectives, not only from technical ones.

# SHIFT VERTICALLY

## CHALLENGES

After clarifying responsibilities and activities of SET, getting decision makers' supports and colleagues' interests, I started actions as SET. Additionally, LINE hired new employees and formed a team of SET. I thought we could proceed our activities more quickly and widely. However, we faced with new obstacles.

At first, we implemented a failure detection system for **public APIs**, however, it didn't become established in the product development team. We implemented test scripts for these APIs, called them via CI servers periodically, and notified errors and failures to the product development team quickly. We utilized Test Automation and CI as a failure detection system. We also used the common technologies like JUnit, Spring Boot, Jenkins, and so on for the product development team. Failure detection worked partially and some developers started implementing them. Although, test scripts written in JUnit were hard to read, implement, and maintain for the Product Manager and most of developers. Additionally, SET team and the product development team have been working at different offices. Our communications weren't sufficient to proceed improvements.

Other obstacle was that performance problems at **Sticker Shop** had emerged. They had been using one in-house performance testing tool. However, it couldn't provide enough capabilities to detect emerging issues. Moreover, they need to write test scripts with groovy, an unaccustomed programming language for them. Therefore, writing test scripts was not fast and effective.

Another challenge was that consulting-style approach didn't work. We often provided guidelines, ideas how to design good test scenarios, and test script examples widely. However, most of colleagues didn't utilize them to improve their testing problems. We needed to find ways to expand ideas and to improve their work more effectively.

## ACTIONS

For achieving our mission, we started working with product development teams deeply to improve their processes. In other words, we started working, learning, and solving essential problems with them.

### REFINE FAILURE DETECTION SYSTEM WITH KARATE

For **public APIs**, we started direct conversations with the product development team members to discover their real needs and concerns at first. In other words, we did “Product Discovery” approach again. We talked daily via video conference system. We discussed with the Product Manager if he came to our office.

Through these discussions, we found that test scripts written in JUnit were hard for them. Therefore, we investigated and proposed lots of testing tools to them.

Finally, we chose Karate [9] framework. It provides features specific to API Testing with BDD (Behavior-Driven Development) style and Gherkin format. It was easy to read, implement, and maintain for both developers and the Product Manager. Especially, defining the preferable state was easy to understand for them.

After decision to use Karate framework, we SETs and the product development team members started rewriting test scripts from JUnit to Karate collaboratively. SETs wrote examples at first. SETs guided "Developer Testing" to achieve and expand "Build Quality In" idea by working with developers. SETs supported solving architectural problems of Karate. After 3 months' collaborative work, finally failure detection system with Karate became established in this product development team.

### IMPLEMENT NEW PERFORMANCE TESTING TOOLS WITH KOTLIN

For **Sticker Shop**, we did the same approach as **public APIs** to discover their real needs and concerns at first.

We found that improving to use the existing in-house performance testing tool was impractical. It could produce only 10% of loads what we wanted to test. It was not easy to expand and/or modify features. Additionally, most of the product development team's members were familiar with Kotlin language. Implementing test scripts with Groovy was hard for them. Moreover, usage of Docker [10] and Kubernetes [11] were expanding at that time in our company. We thought it was a good chance to utilize these new tools and approaches to improve our performance testing.

Therefore, we decided to create a new in-house performance testing tool named "Ayaperf". Ayaperf is a Java wrapper of Locust [12] that can use Kubernetes to increase loads easily with enough volume. Developers can write test scripts of performance testing with Java and Kotlin. We did iterative and incremental style to implement and improve Ayaperf with **Sticker Shop** developers. After 3 months' collaborative work, finally Ayaperf became stable. Developers started detecting performance issues with it before release. Additionally, they could correct issues by themselves without hurting production code. They found and solved 3 hidden performance issues by utilizing Ayaperf.

### IMPROVE PRODUCT DEVELOPMENT PROCESSES AS A HABIT

At **public APIs** and **Sticker Shop**, we found the effectiveness of working with product development teams to find their real needs and solve them. This approach worked well. However, I thought it was not enough and sufficient. I saw that lots of teams stopped solving problems by themselves after coaches left teams. It is a failure if improvements don't continue after coaches' left. Therefore, I expanded our activities to making product development process improvement as a habit especially at **public APIs**.

We had found and solved issues as homework every week. We had continued applying new Karate features and refactoring test scripts. Additionally, we had implemented a notification mechanism via slack to reduce MTTR. Moreover, we had asked product development team members for clarifying objectives, quantitative values they will provide to users, and rough milestones of each task every week. We utilized the idea of Scrum framework to make continuous improvement as a habit of the team. We had continued these activities for about 3 months.

After that, the product development team became able to clarify quarterly milestones, prioritize tasks based on business values, and improve test scripts and the failure detection system by their own. They started decreasing outages dramatically. They really became the self-organized team. Finally, we stopped supporting the team.

## RETROSPECTIVE

We could solve essential problems and improve processes of each product development team by working collaboratively and deeply with them. We SET and product development teams implemented Test Automation and related techniques based on the idea of "Product Discovery". Additionally, each team becomes sophisticated. For example, the Product Manager of **public APIs** writes test scripts with Karate routinely. He often says that the Product Manager may disturb the team by writing production codes, but can contribute to the team by writing test scripts! He is utilizing test scripts to understand behavior of the product deeply, to clarify next actions and goals of the product and the team, and to guide team members doing "Developer Testing" for "Build Quality In".

Additionally, we learned a lot of things to improve our approaches through working with them. The consulting-style approach is useful to keep the whole image of activities, however, we cannot approach essential problems. On the other hand, the working-together approach is effective to discover and solve essential problems quickly, but we may lose the whole image of activities because of too focusing on one product development team. Therefore, we should utilize both styles based on the phase of activities.

The English word "compassion" derives from Latin's "compati", which means "suffer with". We think this is the point of leading and guiding new things like Agile. We SETs and product development teams had been suffering from the same problems by working together. We had considered solutions and solved problems one by one together. These series of activities had constructed real collaborative relationships. Moreover, these relationships had become boosters for adapting to new Test Automation tools and process improvements.

# BECOME TRANSFORMATIONAL LEADERS

## CHALLENGES

## ACTIONS

## RETROSPECTIVE

# LESSONS LEARNED

# WHAT'S NEXT?

# CONCLUSIONS

# AcknowledgementS

Many people helped me write this paper. Many thanks to Kotaro Ogino, Ryotaro Ikeda, and Takeshi Arai for reviewing most of my writing. Joel Tucci, Yasunobu Kawaguchi, and Xuebin Ma also supported the reviewing process. Dai Fujihara and Takao Oyobe always give me a lot of advice as an Agile Coach. I would like to say thanks to Masaya Mori, Yoshihiko Saeki, Kenji Kosaka and Rakuten. They allowed me to proceed the approaches I mentioned on this paper.

At last, special thanks to Jutta Eckstein, my shepherd, and Rebecca Wirfs-Brock. I could not write this paper without their insightful and positive suggestions.

REFERENCES

[1] Transformational leadership. https://en.wikipedia.org/wiki/Transformational\_leadership.

[2] Forsgren, N. Humble, J., & Kim, G. 2018. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. IT Revolution Press.

[3] Fowler, M. 2014. Microservices. https://martinfowler.com/articles/microservices.html.

[4] Nygard, M. 2018. Release It!: Design and Deploy Production-Ready Software 2nd Edition. Pragmatic Bookshelf.

[5] Hussman, D. 2015. Product Discovery On A Single Page. http://productdiscoverycanvas.com/tag/david-hussman/.

[6] SonarQube. https://www.sonarqube.org/.

[7] Whittaker, J. Arbon, J., & Carollo, J. 2012. How Google Tests Software. Addison-Wesley Professional.

[8] Whittaker, J. Arbon, J., & Carollo, J. 2012. How Google Tests Software. Addison-Wesley Professional.

[9] Karate. https://github.com/intuit/karate.

[10] Docker. https://www.docker.com/.

[11] Kubernetes. https://kubernetes.io/.

[12] Locust. https://locust.io/.